INTEGRATING PAVEMENT CRACK DETECTION AND ANALYSIS USING AUTONOMOUS UNMANNED AERIAL VEHICLE IMAGERY

THESIS

Patrick J. Grandsaert, Captain, USAF

AFIT-ENV-MS-15-M-195

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-15-M-195

INTEGRATING PAVEMENT CRACK DETECTION AND ANALYSIS USING AUTONOMOUS UNMANNED AERIAL VEHICLE IMAGERY

THESIS

Presented to the Faculty

Department of Systems and Engineering Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Engineering and Environmental Management

Patrick J. Grandsaert, BS

Captain, USAF

March 2015

AFIT-ENV-MS-15-M-195

INTEGRATING PAVEMENT CRACK DETECTION AND ANALYSIS USING AUTONOMOUS UNMANNED AERIAL VEHICLE IMAGERY

Patrick J. Grandsaert, BS

Captain, USAF

Committee Membership:

Maj Vhance V. Valencia, PhD
Chair

Maj Brian G. Woolley, PhD
Member

John M. Colombi, PhD
Member

Abstract

Efficient, reliable data is necessary to make informed decisions on how to best manage aging road assets. This research explores a new method to automate the collection, processing, and analysis of transportation networks using Unmanned Aerial Vehicles and Computer Vision technology. While there are current methodologies to accomplish road assessment manually and semi-autonomously, this research is a proof of concept to obtain the road assessment faster and cheaper with a vision for little to no human interaction required. This research evaluates the strengths of applying UAV technology to pavement assessments and identifies where further work is needed. Furthermore, it validates using UAVs as a viable way forward for collecting pavement information in order to aid asset managers in their sustainment of aging road assets.

The system was able to capture road photos suitable for semi automated Pavement Condition Index (PCI) processing, however the algorithm resulted in a maximum F-Measure of 40%. This result is low and indicates the algorithm is not sufficient for fully automated PCI classification. Accurately detecting road defects using computer vision remains a challenging problem for future research. However, using Autonomous UAVs to collect the data is shown to be a viable avenue for data collection, and one that is theoretically faster than current methods at freeway speeds.

To my wife, my family, and my friends. I am one lucky guy and without you I am nothing.

Acknowledgements

This thesis would not have been possible without the help and support of many people. I would like to thank Dr. Jacques for taking a chance on letting a civil engineer study UAVs and selflessly giving much of his time and resources. I would not have been able to get off the ground without you. I am also grateful to Rick Patton for helping me select components, answer my questions, and fabricating pieces that were outside my technical capacity.

I would like to thank my committee for their endless help. I am grateful to thank Dr. Colombi for allowing me to use them as a sounding board for my concepts as my research formed. I am obliged to Major Woolley for taking me from zero to hero with computer vision and other programming topics; thank you for showing me how I can be clever. I am indebted to my advisor Major Valencia for showing me how to convey my ideas and taking a risk on a captain with an idea he is passionate about.

I am grateful for all my teachers during my stay at AFIT. I have learned many things about leadership and vastly increased my toolbox for the rest of my career. I would like to thank my classmates for their help and their companionship in these difficult times, this has been my best tour by far and it is in no small part due to our group. I would like to give a special not to my fellow advisees in Major V's afterschool thesis club. We were pretty cool.

I would finally like to thank my family for their support through this time, all times past, and all times to come. Although I am always far away, I always feel you with me. In conclusion, I would like to thank my wife for doing her best in an ever changing life. I love you very much, you make me better than I ever thought possible, and I could not do anything without you.

Table Of Contents

List of Figures

Page

Figure

List of Tables

Page

Figure

INTEGRATING PAVEMENT CRACK DETECTION AND ANALYSIS USING AUTONOMOUS UNMANNED AERIAL VEHICLE IMAGERY

I. Introduction

General Issue

Many forces are pushing for cheaper implementation of current requirements in the U.S. Air Force. The Air Force is currently reducing its active duty force size by 16,700 by the summer of 2015 (Losey 2014). This reduction is not coming with any additional funding for personnel, so the Air Force must continue its current operations with fewer resources. The Vice Chief of Staff of the Air Force General Larry Spencer has been touting programs to find where the Air Force can save money and asking every Airmen "can I do it cheaper?" (Pawlyk 2014). When resources are restricted, choices must be made on what will be funded.

One such choice is funding data collection for infrastructure assessment. Decision making preferences are all subject to hyperbolic discounting which is the preference for present payoffs over future payoffs. A decision maker, then, will tend to make decisions in his immediate best interests rather than future benefits (Laibson 1997, 443-477). With regards to infrastructure, current asset owners value future asset health significantly less than an asset owner in the future. Since measurements to support future asset health can be a lower priority in the present, it may not be accomplished if there are other higher priority requirements according to the current asset owner.

In order to avoid such decisions, cheaper, more cost effective ways of obtaining asset information must be found. This research explores a method of obtaining reliable data on road

conditions via autonomous collection. Reliable data is required to make economical decisions on how to maintain any organization's transportation networks. Furthermore, this method should reduce time demands on current asset managers. Reducing time requirements are important as reductions in the Air Force workforces have reduced time for employees to collect and recollect this data. Operating an autonomous system will reduce time in data collection and will not force decision makers to de-prioritize accomplishing road condition assessments. Developments which make reliable data gathering easier will make data collection more likely to happen and will, in turn, make better decisions more likely in the future. This research explores the use of autonomous systems for data collection on road conditions.

Background

The United States Air Force currently utilizes Pavement Condition Index (PCI) to measure pavement health on its Air Force installations. There are several different methods of accomplishing this index which will be provided and explained in Chapter 2. While these methods do accomplish the requirements of the Air Force, this research intends to explore the area of autonomous systems that may reduce the time and cost required for these assessments with current methods. A major cost to consider in performing PCIs is the time needed to operate the equipment and assess the distresses in the road. This research furthers the potential to reduce the time spent by the human in the loop and decreases the overall time and cost in making road condition assessments.

Assumptions and Limitations

Due to the complexity of the problem and the breadth of systems, this research focuses on only a portion of the PCI process. It will only identify cracks; it will not classify their severity or what type of crack it is. The identified cracks are compared with hand identified crack location

accomplished by the researcher. Secondly, this research is limited to the air space a UAV can operate. Current AFIT UAV research operations are only approved at Camp Atterbury, IN, and include only one stretch of paved road to the north of Himsel Army Airfield. This significantly reduces the sample size available to this research, but is sufficient for a proof of concept. Third, this research is also limited by image sensor technology that restricts the rate of capture and the resolution of the images. Finally, this research is limited to the meteorological conditions present during the flight tests and the number of flights flown.

Scope

In order to further the ability to make collecting reliable data easier, this research asks the following questions:

- How can photographic imagery from small UAVs be transformed into pavement condition information?
- What are differences in performance between automated UAS and manual ground-based methods?
    - What errors are made in UAS pavement crack detection?
    - What is the effective rate of UAS inspection methods?
    - What is the reliability of UAS inspection methods?

This research defines effective rate as *recall*, or how many of the cracks can be identified from the cracks present. Further, this research defines reliability as *precision*, or how many of the cracks detected are actually cracks. These are standards used in information retrieval, have been adapted for computer vision techniques, and are fully applicable for data collection in road condition assessments.

Overview

Following this introductory chapter, this thesis provides a literature review in Chapter II which discusses the relevant, previous discoveries which have led to this systems possibility. Chapter III provides the methodology and provides detail on how the proposed system operates in this research. Chapter IV provides the analysis will which determines how well the system worked against the ground truth. Finally, Chapter V presents conclusions as well as discusses future applications and future research.

II. Literature Review

Pavement health and classification, UAV technology, and computer vision techniques are well studied areas, and there exists a large body of knowledge for these topics. The literature review contained in this chapter will provides a summary of the work in these areas as well as existing work that combines and integrates these topics. This review begins by defining characteristics of a road and how the characteristics determine road health classification. The reader will understand factors that cause roads to deteriorate and how the deterioration appears. The review then transitions to how others have attempted to automate characteristic collection, identification, and classification of road defects. This background information discusses how far the technology has come to achieve its current state of survey road health. Next the review discusses current UAV uses and known applications in civil engineering. Finally, current attempts to use computer vision techniques that could prove useful to this research are discussed. Together, the sections in this chapter give the reader the state-of-the-art status in pavement health classification, UAV technology, and computer vision.

Pavement Health

Pavement repair has been a concern since humans first began building roads. In ancient Roman times, repair decisions were left to a master road builder's expertise in order to determine when a road should be repaired (Federal Highway Administration 2011: 1). This method of repair and maintenance continued until consolidated road organizations began to emerge. One of the first road organizations in North America was formed in 1896 in response to the growth in popularity of the automobile (Haas 2001: 3). These early road organizations were important in developing and standardizing pavement designs and construction standards.

Knowledge and technology in pavements continued to grow with the modern era of pavement management beginning after World War II. The start of the modern era in pavement management came with the American Association of State Highway and Transportation Officials (AASHTO) road tests from 1958 to 1961. The AASHTO road tests defined the characteristics of a road which can be measured and then used to predict service life (Haas 2001: 3-4). The AASHTO road tests were groundbreaking as they led to a common understanding of how road organizations can measure pavement health. Today, pavement health is categorized by four main characteristics: roughness, structural performance, friction and distress. While these principles apply to both concrete and asphalt, the research focuses on asphalt roads. This research focuses on asphalt in order to simplify the computer vision problem to a manageable initial level.

Roughness is a measure of a rider's comfort while traveling down a road. Developed in 1983, and now an established, global standard, transportation engineers use the International Roughness Index (IRI) in order to compare roughness measurements from different locations and roads. (Sayers, Gillespie, and Queiroz 1986: 26). A smoother ride is classified with a low IRI value, while a bumpy ride is classified with a high IRI value.

There are two typical ways to measure IRI. One way is through specially equipped and calibrated vehicles that measure the change in vertical profile of the road. This process can present problems as the results will vary between any two data collection attempts. The measuring vehicle cannot maintain its precise characteristics over repeated measurements, and this prohibits identical values for each measurement (Shahin 2007: 94). A second method measures the road profile with a laser and mathematically computes the roughness index. Measuring the roadway in this direct manner allows for repeatable results. The use of lasers has

become more popular as its relative price has decreased in recent years (Shahin 2007: 94). Roughness is an important consideration for rider comfort; however, it by itself is not useful for optimizing repair funds (Shahin 2007: 93). The IRI will increase as the age and use of a road continues, unless maintenance and repair options are performed.

The second Pavement health characteristic of structural performance is determined by the different layers of road material as well as the two "foundational" layers of the base course layer and underlying soil type. With current technology, the structural strength of a road cannot be measured without being in close physical proximity to the subject. One method of measurement, non destructive testing (NDT), is good for determining strength of a road in its natural environment (Shahin 2007: 61). For asphalt pavements, NDT can determine measures such as elastic modulus of each structural layer, allowable loads for a specific number of load applications, and overlay thickness design (Shahin 2007: 61). NDT measures the deflection of pavements based on known weight, distance, and loading applications. The deflection data is compared to previously determined models which then predict the structural strength of the road. NDT is useful for differentiating different road sections. Separating road sections allows for more accurate modeling of PCI, and allows for specific applications of maintenance and repair at the right time for each section. (Shahin 2007: 79). Properly segmented models allow for a more accurate prediction of a road's PCI decay over time.

The third health characteristic of roads is friction. Friction is the ability of the road to provide a vehicle with a maneuverable surface. Friction on a dry paved road is not a major factor in pavement health; however, friction becomes important in wet or icy conditions as vehicle maneuverability significantly decreases. The primary reason for determining friction information is to identify if a road demonstrates deteriorating skid resistance, which would

indicate decreased safety as an increase in accidents is likely to occur (Shahin 2007: 117). While friction is an important characteristic from a safety perspective, this health characteristic does not affect economical repair prioritization.

The final characteristic is pavement distresses which are classified through the Pavement Condition Index (PCI). The PCI was developed by the US Army Corps of Engineers to develop a repeatable, objective method of recording distress information (Shahin 2007: 17). PCI identifies different road distresses, defines their severity levels, suggests repair methods of these distresses, and identifies possible causes. PCI is solely determined by the distress data (Shahin 2007: 17). Therefore, distress data is extremely important to road service life forecasting. Cracking, rutting, and loss off aggregate on the road surface are different forms of distresses. Traffic loading, fluctuating temperatures, or poor construction are example causes of pavement distresses (Shahin 2007: 43-44). Given that obtaining distress data, and thus PCI information, is so important, this research then looks into different methods of collecting distress data over the years.

PCI Methodology

Unified Facilities Criteria (UFC) 3-270-08, *Pavement Maintenance Management*, governs PCI inspections for asphalt roads. The inspection process begins by breaking roads into similar sections based on its structural composition (thickness and materials), construction history, traffic, and PCI (United States Department of Defense 2004: 2-1). Once roads are broken into sections, each section is further broken down into smaller sample units. Each sample unit represents about 2000 sq ft of the road section. All or some sample units may be evaluated to determine the section's PCI. UFC 3-270-08 determines the number of sample units required based on the number of sample units in a section and the variance in PCIs between the sample

units. Sampling at the rate specified by the UFC yields a PCI for the section that is within 5 points of the true section PCI (United States Department of Defense 2004: 3-5).

The current method to determine a PCI requires a trained technician, a hand-odometer, a 10 ft straight edge and ruler, and a PCI distress guide (United States Department of Defense 2004: 3-2). The odometer is to measure each sample unit size. The ruler and straight edge are to measure the characteristics of each distress since different distresses require different types of measurement. The PCI distress guide is used to identify and verify the type and severity of distress. For example, alligator cracking, one type of distress, is characterized by parallel cracks in the roadway. They are found only in load bearing portions of the roadway and their severity is defined by how interconnected the cracks are as shown in Figure 1, Figure 2, and Figure 3. The PCI distress guide provides these descriptions and visual depictions for the technician during an assessment. Additionally, each distress is given a deduct value based on its percentage of the section and the severity of the distress as shown in Figure 4. This deduct value chart is also part of the PCI distress guide. In using this chart, the sample unit's PCI is determined by subtracting each deduct value from 100. The section's PCI is determined from the weighted average of the inspected samples or the average of all the samples in the section.

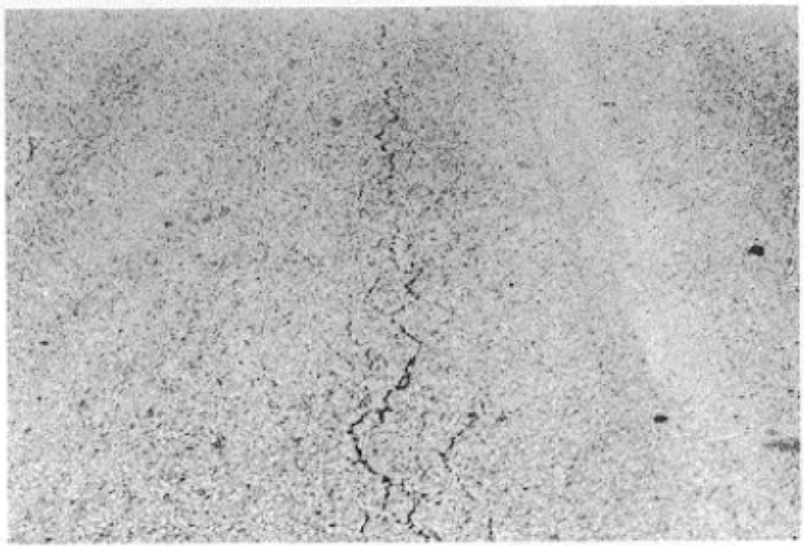

Figure 1 - Low Severity Alligator Cracking

Figure 2 - Medium Severity Alligator Cracking

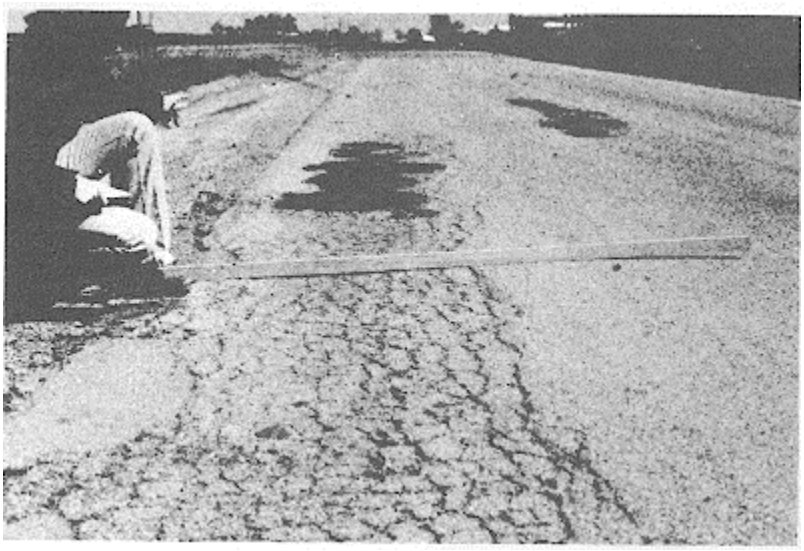

Figure 3 - High Severity Alligator Cracking

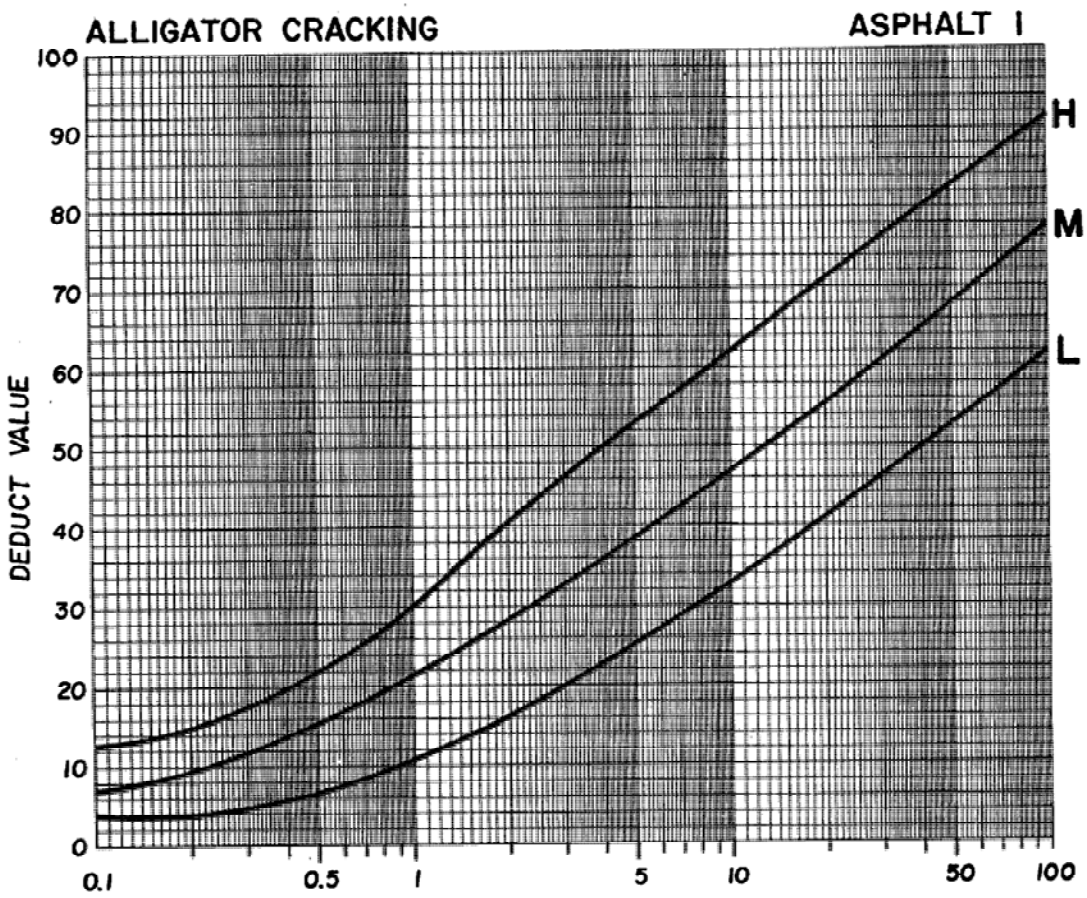

Figure 4 - PCI Deduct Value based on Severity and Distress Density of Asphalt Roads

Variations of performing PCI Inspections: Manual, Semi-autonomous, and Autonomous

There are three current methods for completing PCI inspections: manual, semi-autonomous, and autonomous. Manual distress identifications by personnel on site were the original and only way to accomplish distress identification during early roadway assessments. This practice is still used today by many road organizations which involve a trained technician on-site determining the proper section size to inspect. This method is the most dangerous and usually the most expensive method of distress identification (Elkins et al. 2013, 22). Although this method has shown to have similar results in distress type and quantity as semi autonomous collection, it generally results in lower severity classification (Shahin 2007: 54-55). This could lead to a lower PCI and could lead to a confusing trend if semi autonomous and manual identifications are used interchangeably from year to year. One study has shown manual inspections can cost $\$0.10/yd^2$, which is .01 more expensive than semi autonomous inspections (Cline, Shahin, and Burkhalter 2003: 4).

Semi-autonomous classifications are a second method of pavement assessment where distress data are collected by a system and is classified later by a human. A camera captures visual images or video of the road and an analyst analyzes this capture for defects in the road. This method is available commercially, such as PAVER ImageInspector™, and can reduce the risk to the pavement inspector (Elkins et al. 2013: 22). The variability found between this method and the manual method is similar to variability between different inspectors. Although variability exists in both methods and is an issue of human judgment and interpretation, the assessment reliability is equivalent between the manual and semi-autonomous methods (Shahin 2007: 54-55). In understanding the cost of this method, Clark and Cline, Shahin and Burkhalter

show that this inspection method is marginally more cost effective at $\$0.09/yd^2$. (Clark 2000: 14; Cline, Shahin, and Burkhalter 2003: 4).

Finally, there are fully automated systems that detect cracks in the road. This method can reduce costs, reduce variability between ratings, and reduce man-hours by personnel (Elkins et al. 2013, 22). Currently, commercial companies use laser range data to identify and classify road cracking with 95% accuracy through using specially equipped trucks travelling at freeway speeds (Laurent et al. 2012, 1). Although a human driver is still required to operate the data collection vehicle, this method takes away the need for a human to classify the distresses by automating the data processing and analysis. A separate human analyst at a computer station is no longer needed and therefore cuts human workload significantly. With advances in technology, such as an Unmanned Aerial Vehicle (UAV), a new autonomous method may be a feasible alternative to performing PCI inspections. Unfortunately cost data on this method was not available for this research.

Autonomous Inspections with UAVs

UAVs began shortly following manned flight and have had a long road to its current technological state. Early tests began with unmanned delivery systems beginning as early as 1925 with the use of a gyroscope to stabilize the flight (Coletta 1987: 64). While this technology was not deployed for World War One, development continued through the interwar years and was successfully implemented by both axis and allied forces in World War Two (Keane and Carr 2013, 558). UAV use became widespread during the Vietnam War with 3,435 operational reconnaissance UAV sorties (Clark 2000: 14). UAV use continued in the first Persian Gulf War providing visual confirmation of target damage and improving naval bombardment targeting

(Keane and Carr 2013: 558). UAV technology continued to advance as governments saw the benefits and uses of this unmanned technology

UAVs give governments and other users numerous advantages to manned aircraft and are currently in extensive use in current combat operations. These systems can be attractive to governments because they let casualty conscious governments engage in conflicts that might be otherwise unpalatable (Davis et al. 2014: 11). The loss of a UAV is less disastrous politically than the loss or possible capture of a countries' airman. Furthermore, long endurance times of unmanned systems exceed human capabilities and allow for near constant operation. For example, the U.S. Air Force has enough combat UAV aircraft (i.e., Predator and Reaper aircraft) to maintain 65 combat air patrols indefinitely (Reed 2013). This allows for combat units on the ground with instantaneous aerial information at anytime during an operation. However, these aircraft have been cited by a top Air Force General as useless in an environment where its operations can be interrupted by enemy actions (e.g., jamming or surface to air missiles) (Reed 2013). While this does narrowly tailor the usefulness of drones in military contests, they can still give significant advantages in uncontested airspace. Civilian airspace does not have restriction.

UAVs have been shown to be useful in many civilian applications. UAVs can be used effectively in damage assessment operations in the immediate aftermath of a natural disaster (Tatham 2009: 60-78). Employing unmanned systems prevents humans from needlessly exposing themselves to danger in an unknown environment while still providing first responders important information about the disaster area. In other civil applications, UAVs have also been shown to successfully give detailed estimates in earthwork projects (Siebert and Teizer 2014: 1-14). This application saves the construction team's time and labor by negating the need for extensive survey points for earthwork estimates. The saved time could translate to reduced

project costs, an important consideration in budget constrained government work. Although successful and effective, these applications are waiting on Federal Aviation Administration approval and are purely experimental at this point (Federal Aviation Administration 2015). Further advancements in unmanned aerial vehicles could develop quickly once the legislative groundwork occurs. The application that this research explores is the use of UAV technology to autonomously collect imagery of roads for processing with computer vision techniques

Computer Vision Techniques

Computer vision is the practice of utilizing a computer to process visual information and produce data based on that visual information input. Engineers began studying computer vision as early as 1966, when an instructor assigned a summer project for MIT students to have a computer identify objects in a picture and classify them (Papert 1966). While the assignment was originally scheduled to last for the summer, engineers continued to work on the problem. Since then, engineers have made many advances in gathering information from visual imagery; however, computer vision is still limited with some authors suggesting that the method cannot classify objects at the same precision as a two-year old (Szeliski 2010: 3). Fortunately, engineers have developed techniques that allow some possible object detection.

In using computer vision for crack detection, cracks have certain characteristics that lend themselves for identification. First, cracks generally have low intensity values which allow them to be systematically separated from the rest of the image (Giakoumis, Nikolaidis, and Pitas 2006: 179). This knowledge can be used to automatically find the intensity level of the cracks in the image regardless of the lighting levels. Once these cracks are detected and identified, they are connected by looking at their nearest neighbors. A nearest neighbor is a pixel that is closest in

linear distance to the originating pixel. If a line is short enough, it probably is a false positive and is removed. The cracks can then be quantified and transformed into PCI information.

This chapter gives the reader current status in pavement health classification, UAV technology, and computer vision. It discusses recent attempts to use computer vision techniques on crack detection. This chapter also discusses current applications of UAV technology in civil engineering. Finally, it discusses road health classifications, where it started, the important characteristics in road health classifications, and what methods current practitioners utilize today to assess road health. Pavement health and classification, UAV technology, and computer vision techniques are well researched areas and can be combined in new ways to improve road health assessment.

III. Methodology

This chapter discusses three aspects of the research method. First, the equipment used to collect the road condition data is discussed. Details are provided for the camera characteristics and the unmanned aerial vehicle (UAV) airframe used. Processing the data through computer vision algorithm is subsequently discussed. Finally, this chapter discusses how the data was evaluated for efficiency in order to answer the original research questions.

Equipment

Many pieces of equipment were utilized in this research. The research uses a GE Prosillica 1660C camera with a 50 mm lens and a doubler for an effective lens of 100mm. A doubler is a lens attachment that doubles the effective focal length of the lens. The doubling allows for double the effective zoom of the lens, which is necessary to achieve the correct sight picture of the road from an altitude of 100-200 ft. The camera connects to a computer via a Cat 6 Ethernet cable. The research uses a Next Unit Computing (NUC) computer with an Intel i5 processor, 1.8 GHz Processor, 120 GB Solid state Hard Drive, 8 GB of RAM, running a Linux Ubuntu version 14.1 operating system. The images were captured using the image acquisition toolbox in Matlab® and a script that continuously wrote the images to the hard disk. The research uses a 12-ft Telemaster aircraft running Ardupilot version 3.1 with a ground station laptop serving as the controller. The ground station utilized Mission Planner version 2.6.

Camera Characteristics

There are several camera characteristics that were examined in selecting an appropriate camera for this research. The first characteristic examined is camera resolution. This research requires high resolution road images. Originally, this research planned to use a 16 Megapixel camera in order to maximize the image footprint. However, the maximum frame rate of the

camera is 3.3 frames per second (FPS) which significantly reduces the maximum speed of the UAV to 10 knots in order to capture the entire road. This is a slow collection rate and forces the UAV to operate extremely close to the aircraft stall speed. A smaller 3 Megapixel camera was chosen instead in order to utilize the higher FPS capability. This causes a tradeoff: the smaller camera needs to take more pictures in order to capture the entire road as compared to the larger camera. This tradeoff allows the UAV to operate at a higher speed and results in an overall faster capture rate. The smaller cameras also have a lower minimum exposure time and this smaller exposure time is critical in obtaining clear pictures at higher speeds. Figure 5 illustrates the effect of this tradeoff for seven different cameras; each point is the maximum speed at which a camera can operate with an acceptable blur.

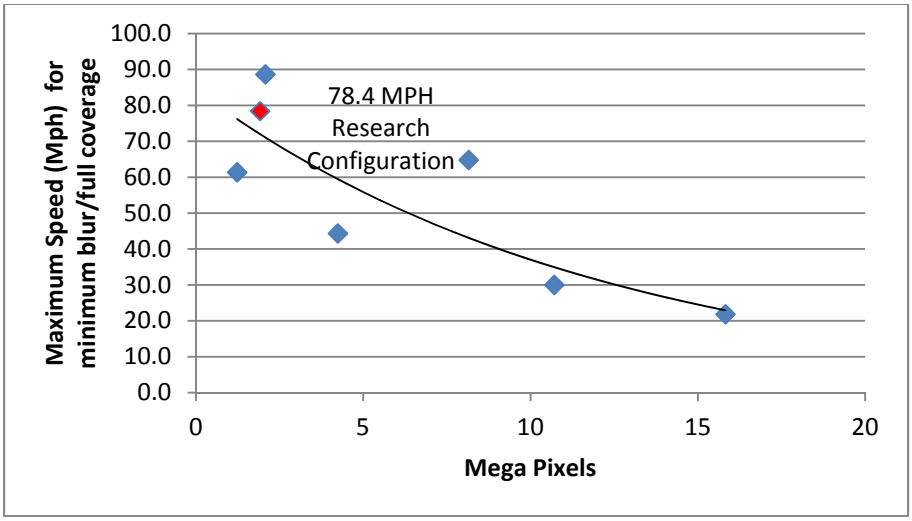

Figure 5: Maximum Speed of Image Capture Characterized by Megapixels of Camera

The next characteristic examined is exposure time and aperture. Exposure time is how long the image sensor is exposed to the outside subject to capture an image. Aperture is the area of light that is exposed onto the sensor at any given time. One way of understanding how a camera system works is to imagine the camera as a 5 gallon bucket under a faucet. The exposure

time is the amount of time the faucet is open; and the aperture is the flow rate of water from the faucet. If the faucet is left open too long, at too high of a rate, the bucket is over filled. If the faucet is not on long enough, or at too low of a rate, the bucket is not filled and the bucket is not usable. The exposure time and aperture must be adjusted to the right levels so the image sensor does not receive too much or too little light. The rate of light streaming in is controlled by the aperture. The standard measurement in photography is uses an f-number. Each time the f-number is doubled, twice the rate of light is allowed onto the image sensor. A camera with a higher resolution will generally require a larger exposure time, and this increased exposure time contributes to blur at higher speeds.

For a picture to be acceptable in the application of crack detection, it must be devoid of blur (Nissen 1993). Blur is calculated in Equation 1 as follows:

$$b = \frac{dwS\cos\phi}{W} \tag{1}$$

where

b = image blur in mm, preferably <0.0508mm
d =exposure duration in seconds
S =subject velocity in meters per second
ϕ = angle between direction of motion and film plane in degrees
w = sensor width in meters
W =field of view in meters.

A minimum acceptable blur in order to achieve quantifiable results is .0508mm (Nissen 1993). The S variable in equation 1 should be maximized so as to cover as much road as quickly as possible but at the same time maintaining a blur of less than .0508mm.

Table 1 shows the maximum speed a camera can move while maintaining coverage of the entire road and realizing an acceptable amount of blur.

Table 1: Maximum Camera Movement while Maintaining Acceptable Blur

	GE 1660C	GT 1290	GT 1910	GE 4900	GX 3300	GE4000	GE2040
Speed (Mph)	78.4	61.4	88.6	21.8	64.8	30.0	44.3
Exposure time (ms)	0.068	0.145	0.055	0.020	0.019	0.018	0.040
Blur at Max (mm)	0.050	0.051	0.049	0.047	0.049	0.050	0.050
Mega Pixels	2	1	2	16	8	11	4
Max FPS	34	33	32	3	14	5	15

In order to capture the entire road, the images must be captured and transferred at high rates into a storage device. A gigabit Ethernet port is rated to 1000 Mb/s; however the researcher's setup tested to only 139 Mb/s. While this is slower than the stated bandwidth, it is still exceeded required testing.

After analysis of the different camera characteristics, this research utilized a GE Prosillica 1660C camera. The camera was selected because of the ability to capture road photos at high speeds. However, this camera required interfacing with a computer instead of storing the data directly onboard the camera. This interface slows down the capture rate (as opposed to storing the data directly on a camera); but this trade-off with the interface allows for sophisticated capturing and a higher storage capacity of photos. This increased capability as a result of the required camera-to-computer interface outweighs the drawbacks of increased power requirements, overall weight, and increased systems complexity. An example of increased systems complexity was the development of non-manufacturer software. The interface required the use of Matlab® in order to command the camera to capture images and then store those images onto the onboard computer. Matlab® provides a good framework and an algorithm was created in order to capture and store the photos from the GigE camera.

Airframe

Multiple airframes were tested 28-29 May 2013 for general airworthiness and potential use for aerial photography in support of this research. Table 2 lists these airframes tested and applicable attributes examined in the selection process.

Table 2: Airframes Considered for Research

	Payload Space	Endurance	Weight Capacity	Engine	Top Speed
Raven	.25 L	1 Hour	10 oz	Electric	10 Km/h
Sig Rascal 110	2 L	1 Hour	4 lbs	Gas or electric	40 Km/h
Telemaster	4 L	2 Hour	15 lbs	Gas	60 Km/h

Table 2 shows the Telemaster, shown in Figure 6, as a suitable option for carrying the mission payload of a Prosillica GE 1660 camera with two, 3 cell 2200 milliAmp-hr batteries and a small Intel Next Unit of Computing (NUC) computer. The payload is shown as an observer looking towards the ground in Figure 7, and looking towards the sky in Figure 8. The Telemaster has the smallest wing loading than any other fixed wing airframe AFIT currently flies. The small wing loading allows it to maintain control at slower speeds and this allows for clearer pictures, as well as more coverage of the road. It also utilizes flaps, which increase the camber of the wing, generating more lift at slower speeds, and therefore lowers the stall speed. Finally, the Telemaster has the largest payload capacity, allowing more weight for increased photo storage capacity or an additional camera. Although a second camera was not used in this research, this configuration could possibly increase the effective camera footprint in future research allowing for more coverage.

Figure 6 Telemaster UAV

Figure 7 - Camera Payload Top

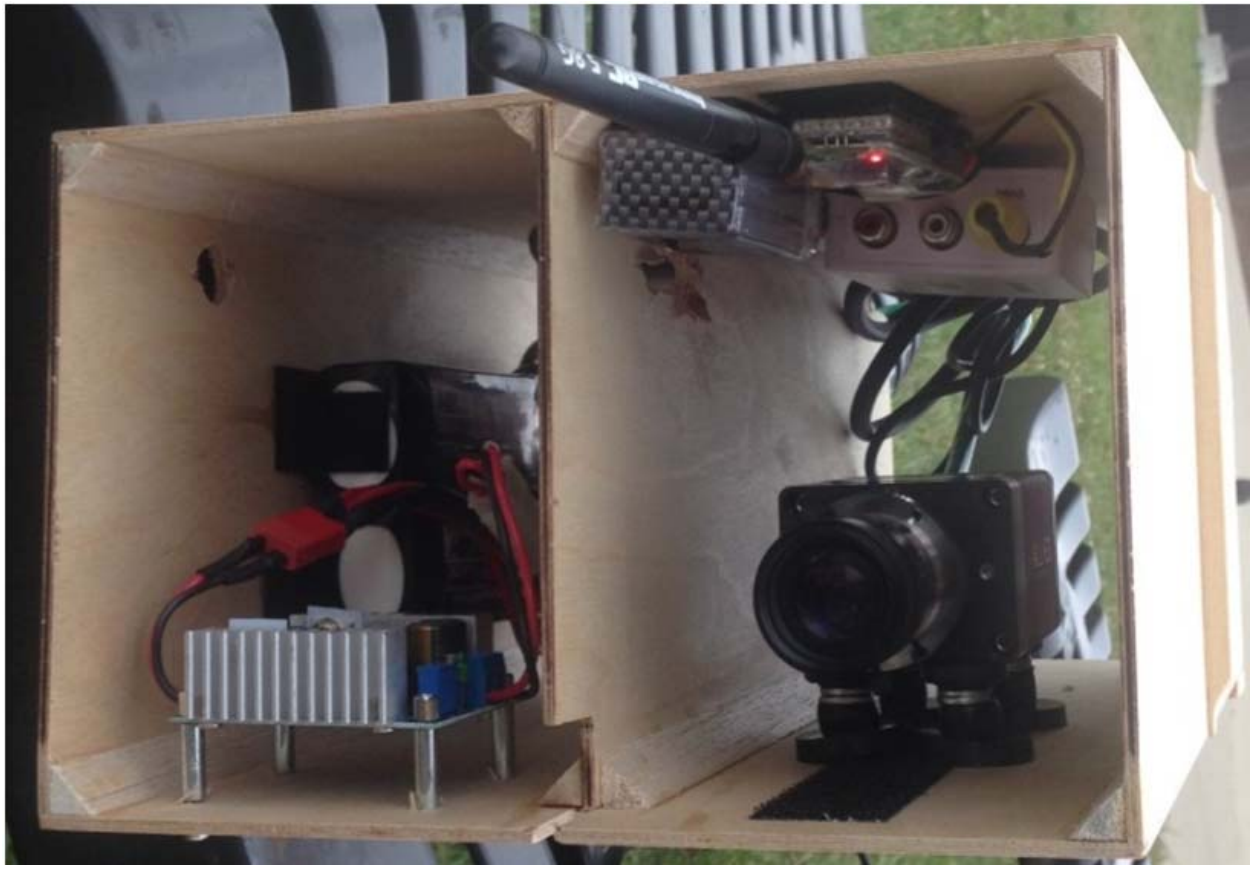

Figure 8 - Camera Payload Bottom

In order to verify the plane was overflying the road, the operating system on the computer was streamed down to a mobile display utilizing a 5.4 GHz Yellow Jacket transmitter/receiver. This allowed the researcher to verify images were being taken of the road while the plane was flying. The entire system was self contained and did not affect the Telemaster's flight apart from being a weighted payload.

Computer Vision

Computer vision techniques are required to identify the cracks from the images obtained. Thresholding is the process of converting pixel intensity values, which can range from an 8-bit integer to a vector of 32-bit integers, to a binary value of one or zero (Szeliski 2010: 127-128). A value at the top of the spectrum occurs when the sensor receive the maximum amount of light

possible; a value at the bottom of the spectrum occurs when no light is registered by the sensor. This is useful for capturing which pixels are part of a crack; but depending on lighting conditions, the thresholding level may vary from image to image. . A solution around this problem is to ascertain the different intensity levels immediately around a given pixel since these levels should be consistent across nearby pixels. Under this assumption, pixels in an image can be in one of the four possible locations as shown in Figure 9. If there is no difference in the intensity levels around a pixel, such as locations "A" or "D" in Figure 9, a pixel is likely within a crack or outside of a crack; if there is a large difference, such as at locations "B" or "C" in Figure 9, the pixel is likely the edge of a crack (Zou et al. 2012: 231). For grayscale images, the algorithm determines the surrounding intensity level of each pixel to determine which intensity to threshold the image.

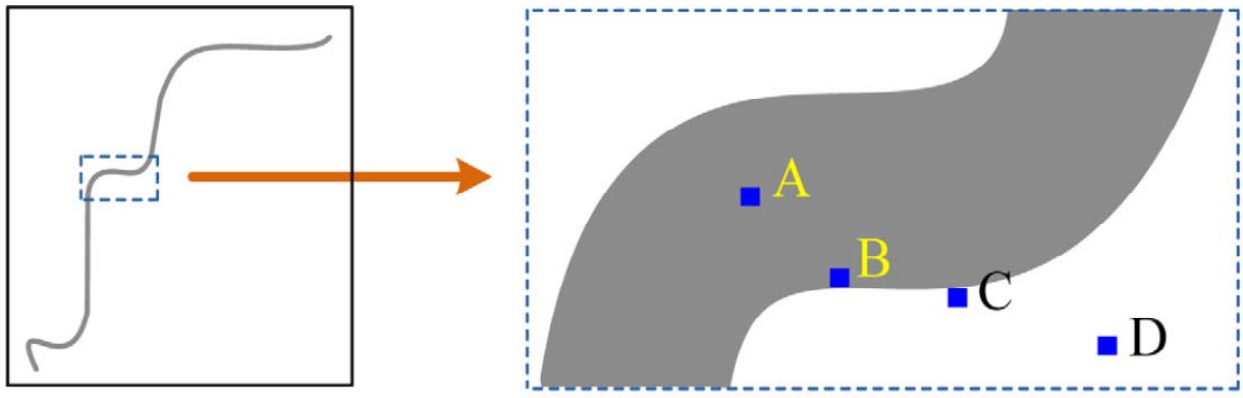

Figure 9 - Pixel Locations Near Cracks (Zou et al. 2012b, 231)

The intensity level of a pixel that has low to no difference between itself and its neighbors is the likely intensity of crack pixels in a photo and is the intensity level that the picture should be thresholded at. In practice, thresholding clears a large portion of the noise in an image, but the algorithm needs additional methods to further reduce false positives. Once the

cracks have been identified, gaps in the detection should be connected if they are reasonably close, while reducing and false positives. The crack pixel locations are then connected to other crack pixels within 40 pixels of it as an initial estimate of how far pixels could be from each other and still be part of a connected crack.

In order to obtain accurate defect data, crack pixels that are close to each other need to be connected while false positives need to be removed. The data was arranged using a graph by plotting potential edge pixels and then allowing edges to be connected between near pixels. This research initially attempted to use an algorithm that compared each point to each other point, drawing a line between pixels only if the distance was below 40 pixels as shown in Figure 10.

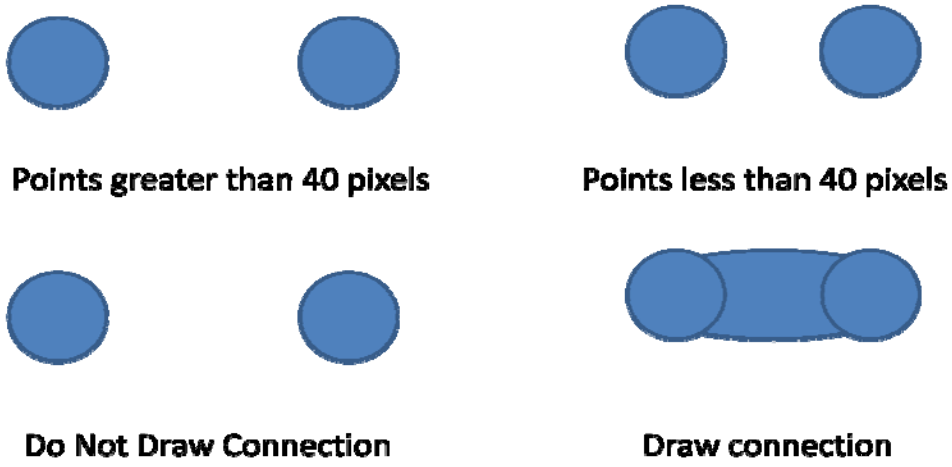

Figure 10 - Connection Query

However, the graph typically held fifteen thousand points and processing time was taking upwards of 50 minutes. A KD-tree method can be used to reduce this processing time. The KD-tree is a popular method of indexing multi-dimensional search trees (Szeliski 2010: 233). The method splits the search area into four separate areas, checks which area meets the search criteria, separates that area into four quadrants and continues until the final value is found or comes to the conclusion that there is no point that meets the search criteria (Woolley 2014).

Processing time was cut to one minute using this method, significantly improving performance. This provided a viable way forward to processing future images.

In order to further reduce false positives, the method must then take into account how to remove extra offshoots, which can be obtained from a minimum spanning tree. A minimum spanning tree takes a fully connected group of nodes and edges and finds the path that connects all nodes for the least amount of total distance. There are two major algorithms for completing a minimum spanning tree: Kruskal and Prim-Jarnik (Woolley 2014). The researcher chose to implement Kruskals algorithm as a readily open source program was available. Kruskal's algorithm proceeds by choosing the smallest distance edge and retaining it only if it connects a new node. Edges connecting already visited nodes are removed. Once all nodes are connected, the algorithm is completed.

To evaluate success of the algorithm, the two standard measurements of accuracy in computer vision (precision and recall) will be used. Recall is the ratio of the number of true positive identifications the algorithm found over the actual number of identification possible (Ting 2010: 781). Precision is the number of true positives found by the algorithm over the total number of positives found by the algorithm (Ting 2010: 781). These two measures account for two errors an algorithm might produce and can be combined into an F-measure. Precision, recall, and F-Measure are calculated with the following equations:

$$\text{Precision} = \frac{\text{True Positives Found by Algorithm}}{\text{True Positives} + \text{False Positives}} \quad (2)$$

$$\text{Recall} = \frac{\text{True Positives Found by Algorithm}}{\text{True Positives} + \text{False Negatives}} \quad (3)$$

$$\text{F-Measure} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \quad (4)$$

These equations will be used to calculate the error and success rate of the algorithm against the ground truth. Ground truth is established by the researcher hand marking the cracks in each picture. Each pixel that is identified as a crack and is within 15 pixels of a hand identified crack will be considered a true positive. 15 pixels was chosen to make sure pixels that are reasonably close will be counted as true crack pixels. Any pixel identified as a crack and not within 15 pixels will be considered a false positive and added to the denominator of the precision calculation. Any hand identified pixel that is not within 15 pixels of an algorithm identified pixel shall be added to the denominator of the recall equation. The F-Measure is important as it is the harmonic mean between precision and recall. It allows the algorithm to be evaluated while accounting for false positive and false negatives, giving a true understanding of how accurate the algorithm is. A human would identify cracks at an F-Measure of near 100; an algorithm shall be considered successful if it achieves a F-Measure of at least 95.

Finally, one last technique will be used in order to achieve a more accurate result: histogram equalization. Histogram equalization involves spreading out the intensities of the grayscale images to the extremes of the possible ranges. If an images darkest pixel intensity value in is 50 and its brightest intensity value is 150, each value will be changed according to its relative position in the original image. Each pixel intensity will subtract the image minimum, be multiplied by 255 and divided by the length of the range of intensity values in the original images. This is attempted in order to widen the differences between the crack pixel intensities in the image and the non crack pixel intensities. The above methods will be attempted and the results will be discussed in Chapter 4.

This chapter discusses how the efficiency of the algorithm is determined in order to answer the original research questions. It describes the techniques used to process the road

images into road distress data. Finally it described the necessary characteristics of the payload in order to capture road images. This chapter reviewed all the components of the methodology and how it was used to collect data.

IV. Results

Two days, August 22nd and October 30th, of UAV flight occurred at Camp Atterbury, Indiana. Both days were partly cloudy and 71 and 53 degrees, respectively. The flight path of the UAV did not always fly directly above the road, and the images did not always contain the road. Way points were adjusted north and south of the road to attempt to fix this issue, as well as differing distances between waypoints, but an exclusive straight-line flight over the road was not achieved. The exposure time and gain settings were varied before each flight in order to obtain optimal pictures, however there continued to be visible differences in the lighting conditions of the images.

Figure 11 – Road Analyzed

The imagery collected ranged from ideal to unusable depending on the flight and the lighting conditions. There were a significant number of usable images, and the results of processing some images are shown in Figure 12.

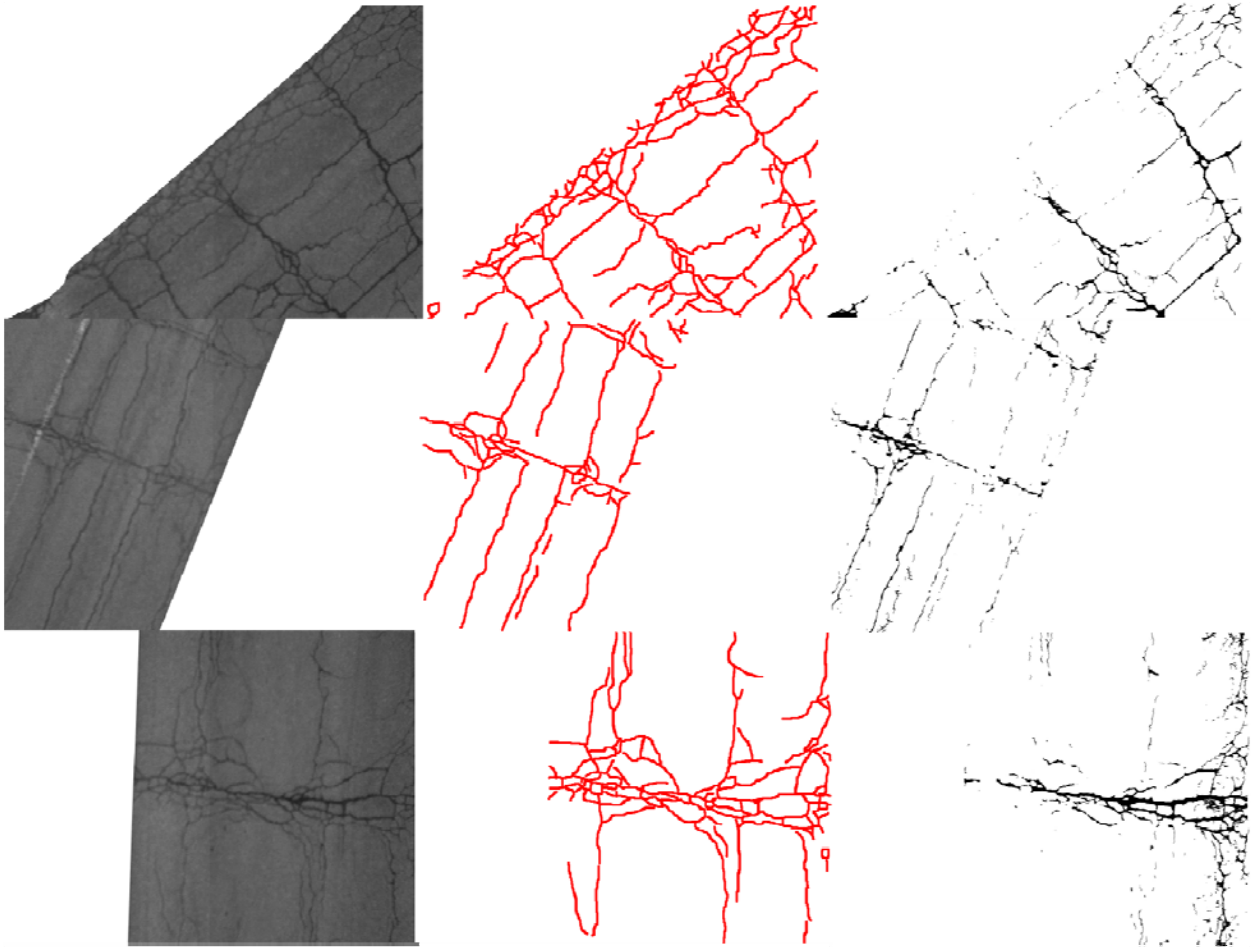

Figure 12 - Captured, Ground Truth, and Processed Images

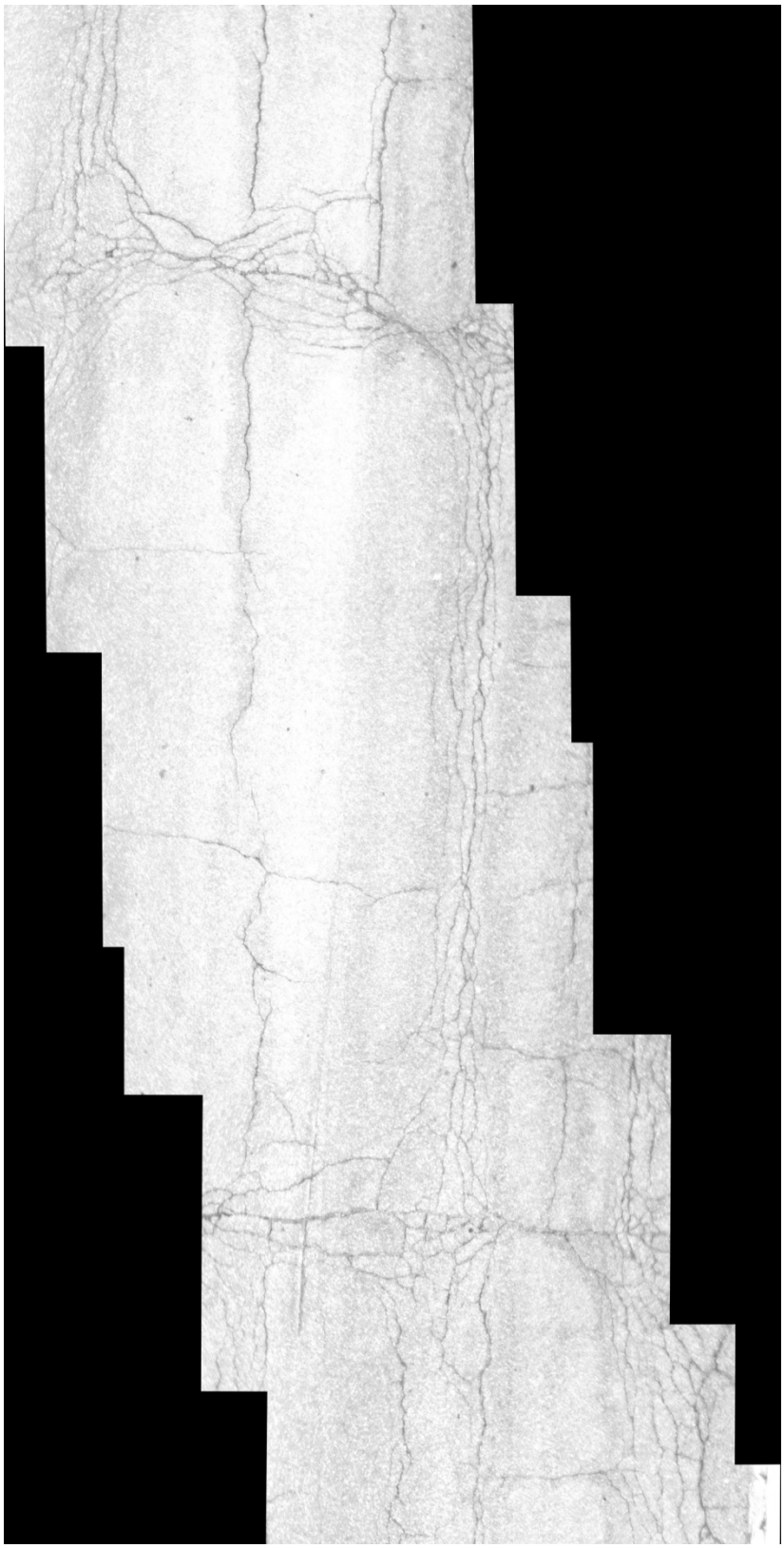

Figure 13 - Combined Road Photos

The road was 2000 feet long, as shown in the yellow line in Figure 11, and just within the researches approved airspace. The road was chosen because it was the only possible road to survey; however, it was an also an ideal road for research with many sections of different distresses, as well as some without distresses at all. The original intent was to combine each road photo together in order to capture the complete section in one image. However, the flight path of the UAV did not directly follow the road and led to incomplete coverage as shown in Figure 13 - Combined Road Photos

The algorithm to find the optimal threshold intensity did not work at the found intensity. This was compensated by subtracting the found value by the intensity shift in the x-axis of the figures in this chapter. Due to the algorithm finding too many potential crack pixels, the algorithm could not process all the images at the specified intensity level. At too high of an intensity level, the computer is overwhelmed with potential positives within close proximity, and thus highly likely false positives. These images are thrown out, and those images remaining are reflected in the analysis and figures in this chapter.

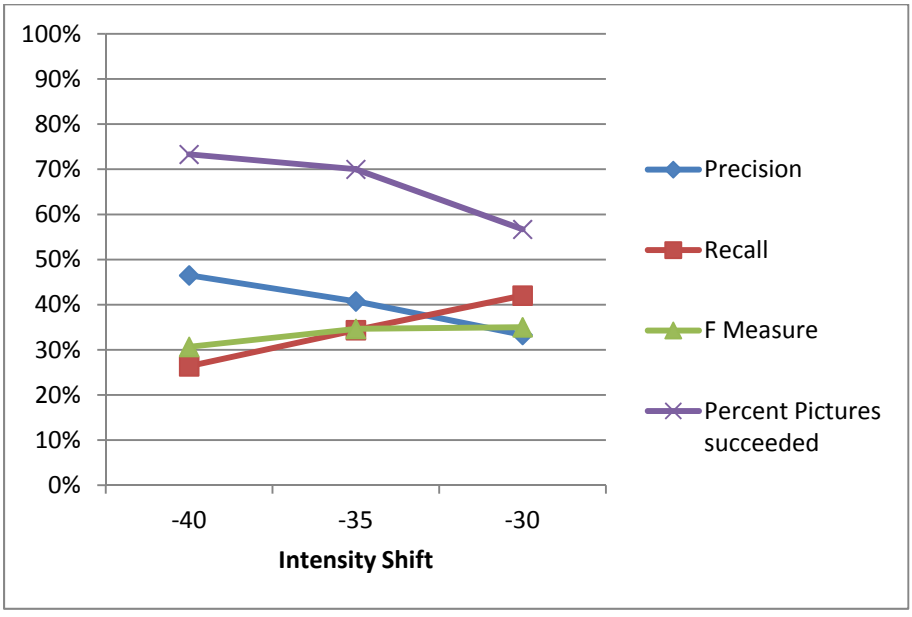

Figure 14 - Precision, Recall, F-Measure of Initial Algorithm

The following are the results of analysis of different variations of my initial algorithm. Figure 14 shows the statistics on the original attempt to identify defects in the road. The highest F-Measure achieved was 35, thus further methods were required to improve the effectiveness of the algorithm. Figure 14 shows the results of attempting to account for the different lighting situations by utilizing histogram equalization. Figure 16 shows the results when the algorithm was adjusted for finding the nearest neighbor using a set true ground distance rather than a set pixel difference. Finally, Figure 17 shows the results of processing a single flight with the clearest lighting setting.

Figure 14 shows a random sampling of 30 total images; the 30 images were chosen at random from a pool of all available road photos from each flight. The F-Measure reached a peak at 35 minus the intensity level found by the initial algorithm. This indicates the original algorithm is either insufficient or the method of capturing images is insufficient. An acceptable F-Measure would be 95 percent as this is the rate expected from a human classifier. Since the F-Measure is lower than an acceptable, further efforts were made to increase F-measure. The next effort was to equalize the intensity levels across the image using histogram equalization. Recall that histogram equalization is the attempt to bring the intensity levels of the image to be consistent with all portions of the image. This process ensures an overly dark portion of the road is the same color of the rest of the road and is not mistaken for a crack.

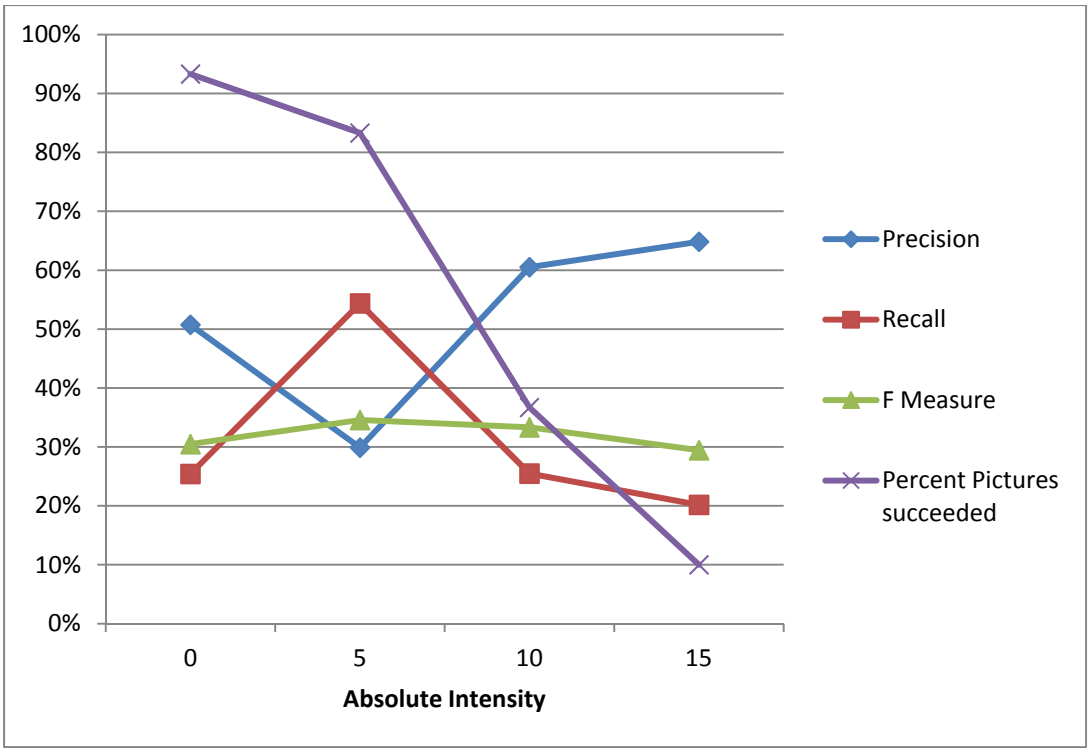

Figure 15 - Precision, Recall, F-Measure of Histogram Equalization Variation

Originally, the images were run at the same intensities as the previous trial; however this led to a success rate of 6.7 percent. The images were instead run from different intensities starting at the bottom of the intensity scale. This led to the results as shown in Figure 15. The F-Measure is still below 95 percent and did not make a significant difference when compared to the original algorithm results. Further changes are needed to produce an effective F-Measure.

Next, the images were processed according to a foot as measured in the picture rather than a set pixel distance for each pixel. The research attempted using true foot measurements in order to increase the recall of the images by including more possible pixels in the higher resolution images, and reducing errors in the lower resolution images. The true spatial distance increases how far the algorithm will look to connect crack pixels on high resolution images, and decrease the distance for low resolution images. The results on this final analysis are shown in Figure 16.

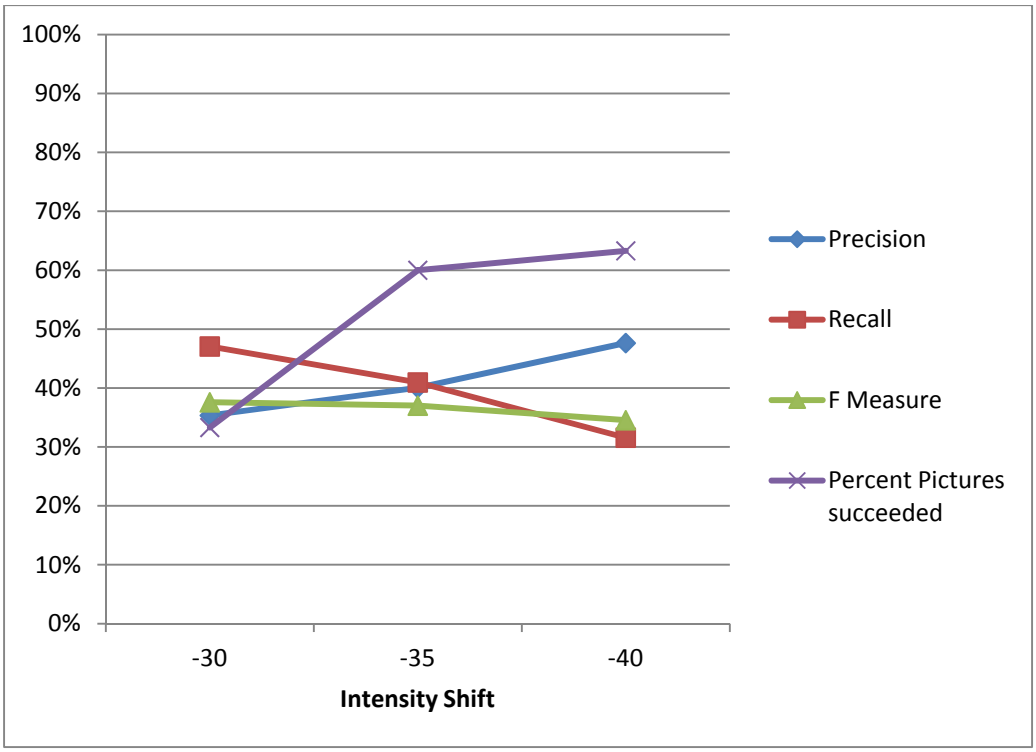

Figure 16 - Precision, Recall, F-Measure for True Ground Spatial Distance Variation

Again, the analysis conducted for Figure 16 did not conclude in an acceptable F-Measure. None of the previous methods achieved an acceptable rate of success, so one final test was conducted to see if the poor numbers were obtained due to different lighting conditions. The results of the final test are shown in Figure 17 . The figure shows the algorithm results from a single flight where lighting settings were the same, as well as relative ground spatial distances. While the precision was much higher for each of these pictures, they captured far fewer true crack pixels as is shown in the low Recall, and thus obtained similar F-Measure values as the previous methods.

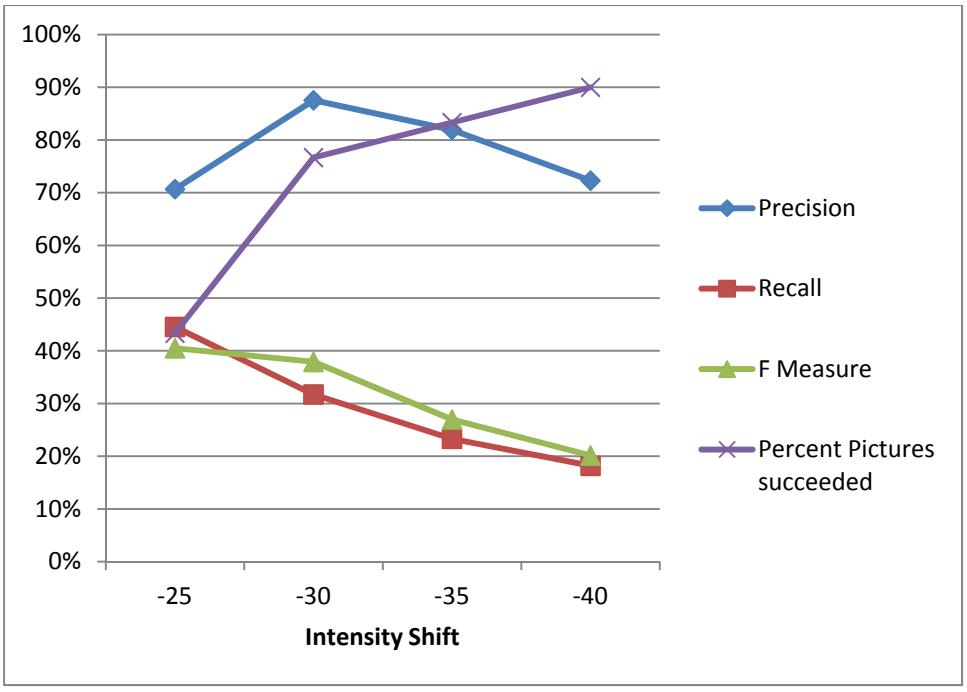

Figure 17 - Precision, Recall, F-Measure for Isolated Flight

This chapter discussed the results of the data collection effort and the effectiveness of the algorithm. Road images were captured at an acceptable resolution with minimum blur, and thus acceptable for semiautonomous inspections. However, the original algorithm and all subsequent variation did not prove useful enough for immediate application. The final chapter will discuss the impacts of these results.

V. Conclusion

This research has shown that photographic imagery can be obtained for road analysis from a UAV. The imagery was clear and capture rate fast enough to cover the entirety of the flight path. Each component functioned quickly to receive and store the visual imagery below it. The imagery could be used in a semi autonomous pavement inspection system with a human operator hand identifying the distresses in the road. While the aircraft was under the eye of a safety pilot, it could perform the survey run on its own. This means surveys could be performed without human interaction, significantly saving on labor costs, assuming purely autonomous operation becomes a lawful activity for UAVs.

While the results show the current system devised for this research is not ready for operational use, it does shed light on how to further this technology in order to obtain reliable data in the future. Once this data and accompanying algorithms achieve an acceptable error rate, the results can be used to transfer PCI information into the Paver database. The Paver database is an infrastructure asset management system used by the Department of Defense in managing its roadways and other paved surfaces. This information allows decision makers to save money by extending the life of existing road assets optimally with well-timed repair options. Such a method of management would not be possible without reliable data that is consistently documented. Figure 18 shows how injecting money on preservation at the right time is crucial to avoiding costly repairs in the future. In a fiscally constrained environment, such information makes the difference between a perpetually failed transportation network, with increased costs for road users, and a stable road network with a condition state that can continue indefinitely.

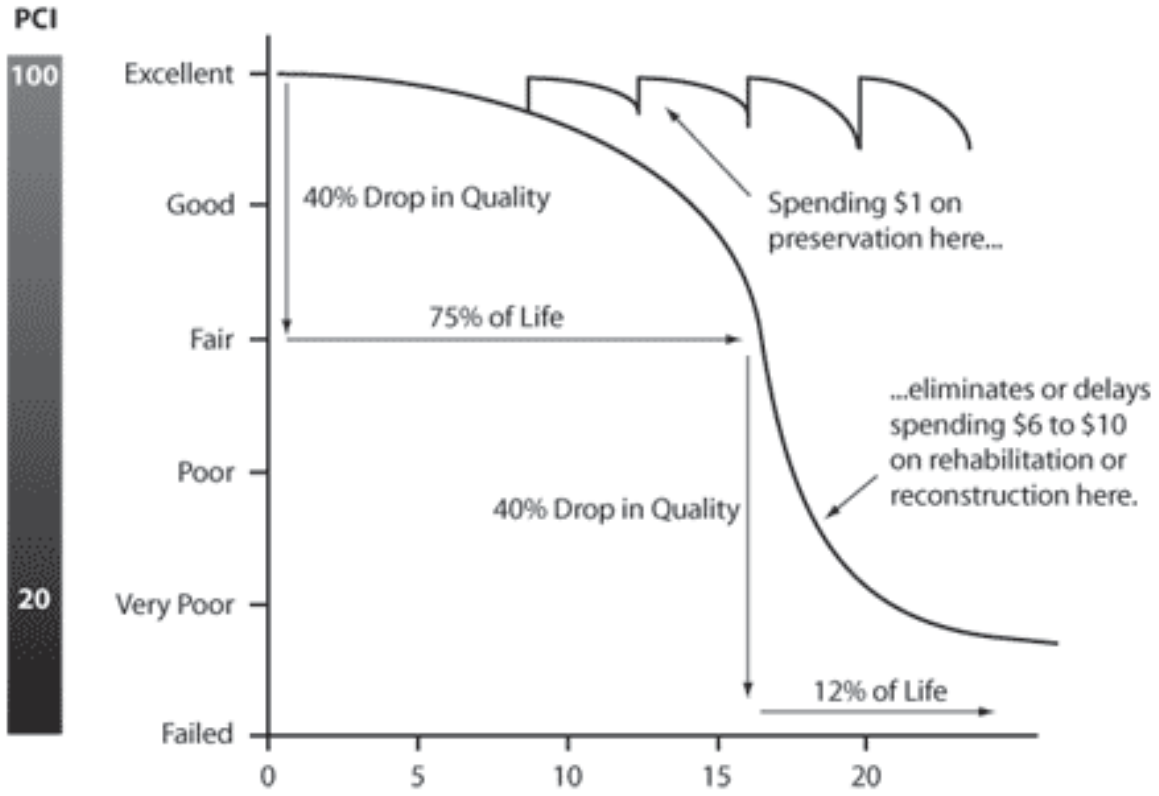

Figure 18 - Optimal Use of Repair dollars (Galehouse, Moulthrop, and Hicks 2011)

Photographic imagery can be obtained by attaching high performance cameras to small UAVs and these systems can be flown autonomously using GPS and fixed waypoints. These images can be transformed into crack data using algorithms, although they will have to be improved for future use. While the current performance is not quite up to acceptable standards, further advances and techniques and be developed in the future that will bring this technology on par with human recording of pavement condition data.

Errors occurred due to sub optimal lighting conditions. The algorithms detected the errors as cracks at one extreme, and missed cracks completely at the other extreme. The current proposed system has objective errors that create an unacceptable product for current uses, but appear to have promise if the algorithm can be improved. The pictures of the road varied from flight to flight due to lack of system self-calibration to account for changing environmental

lighting conditions. Finding a way to control for environmental lighting conditions would allow for the system to more consistently capture road images. This may help future crack finding algorithms and would definitely help humans performing semi-autonomous inspections. This research took photo information at a speed of 25 mph. However the camera system can theoretically accommodate up to 78 mph, as shown in Figure 5, and the Telemaster airframe can fly at a maximum of 50 mph. Accounting for this possibility in increased speed could lead to faster data collection than current commercial methods for residential roads as well as being comparable in speed to assessments for highway roads. However, as shown in the analysis, this method of data process is not yet at an acceptable level for application. A level of 95% for the F-Measure would allow for immediate application.

Limitations

While this technology is promising, there are some key limitations. The autopilot algorithm did not directly follow the road. This may be due to the environment the research was forced to operate under: The flight path of the UAV was a tight circuit in order to maintain operation within the approved airspace. The ideal flight path would have been directly over the road and for a longer, straight line flight, as would be necessary for a real life application of this method. Also, the lighting characteristics of each flight were sufficiently different to make it difficult to achieve consistent results. This could be solved by incorporating a light meter and using it to set the photography settings in pre-flight or during flight. These lighting settings could be done on the ground or even attached to the aircraft to send periodic updates to the camera settings. Next, the crack identification is not at an acceptable level to replace human interaction. This is crippling to further application of the system and must be addressed by further improving crack identification as well as crack classification. Finally, the methodology does not address

other distresses such as rutting, bleeding, or patching. Further research into how to reliably collect three dimensional information and rutting and bleeding from a UAV is required to solve this inconsistency.

Further research

This research brings on many areas for future development. Research can be accomplished on whether this imaging package should be attached to municipal or government vehicles that already cover the entirety of an agencies road network. A study could accompany it with how long would it take a passive system to survey the entire network. Further research could determine the optimal number of vehicles in order to finish a network survey in a reasonable time. Additionally, more research is needed in how to obtain rutting dimensions, or three dimensional data on a road with enough precision to detect rutting of a road from an aerial vehicle. Another area to be addressed is research into how to properly classify different cracking into various categories defined by the PCI process via an intelligent algorithm. Finally, future research should also address how an autopilot system can be better programmed and manipulated to have a consistent ground track over a transportation network.

The overall goal of this research is to push the envelope into faster, cheaper, and better ways to gain useful information on infrastructure. With this information, infrastructure asset managers can make better decision on where to spend their next repair dollar. While the system explored in this research is not ready to accomplish real world application, it is the researchers hope that the research pushed the envelope of knowledge to further discoveries.

Appendix A. Pseudo code

1) Set intensity shift and filenames of images
2) Run through each image from UAV
    a) Create initial result as an red image
    b) Call Plot Script
        i) Load UAV image as grayscale
        ii) Determine brightest possible pixel
            (1) Take the intensity of each pixel and subtract it from the center pixel
            (2) Sum differences
            (3) Add result to intensity array at the index value of the center pixel's intensity value
        iii) Find index that contains the intensity arrays max value
        iv) Set index as the value to threshold the image
        v) Adjust threshold value by previously identified shift
        vi) Threshold image
            (1) Loop through image
                (a) set any pixel below threshold value to 0
                (b) set any pixel above threshold value to 255
        vii) Loop through image
            (1) Create node on undirected graph for each location of each pixel with a value of 0
        viii) Create KD-tree from nodes
        ix) Find pairs of nodes within specified distance using KD-tree
        x) Create edges of graph from pairs

xi) Create a minimum spanning tree (MST)from nodes and edges using kruskals MST algorithm

xii) Remove any nodes that are not connected or only have one edge connected to it

xiii) Plot Graph as the same size as the original image and save as .jpg, overwriting the red image that was originally created

3) Create text file (treated as a csv)

4) Loop through the number of images to be processed

   a) Load algorithm result image

   b) If image is red, skip to next iteration of loop

   c) Print file number in text file and run compare script

      i) Load algorithm result image

      ii) Load ground truth image

      iii) Check every ground truth crack pixel against the corresponding pixel of the algorithm result image within a specified range.

         (1) The loop counts all true positives found by the ground truth and the number of crack pixels in the ground truth file.

      iv) Checks every algorithm truth crack pixel against the corresponding pixel of the true crack image within the specified range.

         (1) The loop counts all false negatives

      v) Calculates Precision, Recall, and F-Measure of each picture

      vi) Prints Precision, Recall, and F-Measure to text file

5) Writes percentage of files processed out of possible images processed

If the algorithm had successfully accounted for 95% of the cracks present, the found crack pixels from the road images would be divided by the total road pixels. This percentage would be compared to the PCI curves for each distress type, which combined with the severity, would yield the deduct values for the road section. The branch ID and similar information for each section would be determined from the GPS tag in the EXIF info tag in the photo. This information would be format into a format that can be easily exported into PAVER.

Appendix B. Code

```
#Test1.py

#File to run Crack Detection Algorithm
#import Thesis code files
import compare5
import plot21

#import libraries to process images
from PIL import Image
import cv2

diff=40#the value the intensity is shifted.  This value is changed for each
run ofthe script

#Different File names for the different variations of the algorithm that were
run
#name='-'+str(diff)+' histogram '
#name='-'+str(diff)+' histogramstatic '
name='-'+str(diff)+' run2normal '#the last file name for processing images
from one flight

for x in range(1,31):

    #creates a red image for the algorithm image.  crack identification
algorithm
    #is successful, the red image is overwritten.  if the algorithm runs into
memory issues
    #the compare script skips processing the image
    im=Image.new("RGB",(1600,1200),"red")
    im.save('C:/Users/Pat/documents/google drive/temp/algocrack/'+str(name)+
str(x)+'.jpg')

    plot20.plot(name,x,diff)

#opens csv for each iteration.  The csv is evaluated in excel
f = open('effeciency'+str(name)+'.txt','a')
count=0
total=0

#compares each algorithm image produced with each ground truth image
for x in list:
    total+=1
    im3=cv2.imread("C:/Users/Pat/documents/google
drive/temp/algocrack/"+str(name)+str(x)+".jpg",1)

    #if image is red, algorithm image was not created, so the image is
skipped
    if im3[1&1].any()==[0&0&254]:
        continue
    f.write('File '+str(x)+'\t'+compare5.compare(x,name)+'\n')
    count+=1

#writes percentage of 30 images that were successfully processed
f.write('& comp\t'+"{:.1%}".format(float(count)/float(total))+'\n')
```

```
f.close()
```

```
#Plot21.py
#This Function takes the filename of the image to be processed as well as how
much to shift
#the intensity level from what the algorithm calculates to be the brightest
crack pixel
def plot(name,file,thresh):

    #Imports to be able to draw a node
    import networkx as nx
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    import matplotlib

    #Imports numberical tools and arrays
    import numpy as np
    from scipy import spatial

    #Imports computer vision tools
    import cv2

    #Tools to determine how long each step takes for operator awareness
    import sys
    import time

    #Image Loading Tools
    from PIL import Image

    #Load image as grayscale
    #Takes File from function call in order to load the correct image
    gray=cv2.imread('C:/Users/Pat/documents/Google
Drive/temp/roads2/'+str(file)+'.jpg',cv2.CV_LOAD_IMAGE_GRAYSCALE)

    print 'C:/Users/Pat/documents/Google Drive/temp/roads2/'+str(file)+'.jpg'

    print "Loaded Image"

    #Loaded Image to show user it loaded correctly
    '''
    namedWindow('dst_rt')
    #cv2.resizeWindow('dst_rt', window_width, window_height)

    imshow('dst_rt', gray)

    waitKey(0)
    destroyAllWindows()
    '''

    #Determines Image Dimensions,
    width,height = gray.shape
    width-=1
    height-=1
    gradient=[]
    pair=[]
    findthresh=[]
    threshold=0
    maxT=0
```

```
    intensities=[0]*256

    #Start Clock to Determine how long this step takes
    begin=time.time()

    gray[0,0]=0
    gray[0,height]=0
    gray[width,height]=0
    gray[0,height]=0
    # Ctrl Q to comment and uncomment blocks
    #Runs Algorithm to determine brightest crack pixel
    #loops through each pixel in loaded image and determines its brightness
relative to its neighbors
    #Positive result indicates it is a darker pixel compared to its neighbors
    #each result is added to an array index from 0 to 255, the array index is
determined by the intensity of the center pixel
    for j in range( 1, (height-1)):
        for i in range( 1, (width-1)):
            pixelintesity=0
            pixelintesity=pixelintesity+int(gray[(i-1),(j-1)]-gray[i,j])
            pixelintesity=pixelintesity+int(gray[i,(j-1)]-gray[i,j])
            pixelintesity=pixelintesity+int(gray[(i+1),(j-1)]-gray[i,j])
            pixelintesity=pixelintesity+int(gray[(i-1),j]-gray[i,j])
            pixelintesity=pixelintesity+int(gray[(i+1),j]-gray[i,j])
            pixelintesity=pixelintesity+int(gray[(i-1),(j+1)]-gray[i,j])
            pixelintesity=pixelintesity+int(gray[i,(j+1)]-gray[i,j])
            pixelintesity=pixelintesity+int(gray[(i+1),(j+1)]-gray[i,j])
            intensities[gray[i,j]]=intensities[gray[i,j]]+pixelintesity
        if j%100 == 0:
            print "......"+str(int(100*(float(j)/float(height))))+"%\r",

    #Ends Clock and prints time to determine how long step took
    end=time.time()
    print "step time: " + str(int(end-begin))+" seconds"

    #print intensities
    # Bar Chart
    #fig, ax = plt.subplots()

    n_groups=len(intensities)

    index=np.arange(256)

    bar_width=.1
    opacity=4
    error_config={'ecolor': '0.3'}

    #rects=plt.bar(index,intensities,bar_width)
    #plt.show()

    max=0

    #Finding brightest probable crack pixel by finding the intensity with the
highest difference from its neighbors
    begin=time.time()
    i=0
    for i in range(256):
```

```
    if intensities[i]>max:
        max=intensities[i]
        threshold=i

end=time.time()
print "step time: " + str(int(end-begin))+" seconds"
print threshold

#Adjusting threshold by value of the function call
threshold-=thresh
pointmatrix=np.array
begin=time.time()

#turning any pixel below threshold white and all others black
for j in range( 0, height):
    for i in range( 0, width):
        if gray[i,j]>threshold:
            gray[i,j]=0
        else:
            gray[i,j]=255

end=time.time()
print "step time: " + str(int(end-begin))+" seconds"
print "thresholded image"

"""
Checks for debugging
namedWindow('dst_rt', WINDOW_NORMAL)
#cv2.resizeWindow('dst_rt', window_width, window_height)

imshow('dst_rt', gray)

waitKey(0)
destroyAllWindows()

"""

G =nx.Graph()
#Creating list of crack pixels
pos={}
k=int(0)

for j in range( 0, height):
    for i in range( 0, width):
        if gray[i,j]>0:
            gray[i,j]=int(255)
            pos[k]=((height-j)*(-1),(width-i))
            k+=int(1)

    if k%100 == 0:
        print "......"+str(int(100*(float(j)/float(height))))+"%\r",
#print pos[0]

print "created crack pixels"

#creating nodes from crack pixels
G.add_nodes_from(pos.keys())
```

```
    print "Created Nodes"

    nx.draw_networkx(G,pos,with_labels=False)
    #plt.axis('off')
    #plt.show()
    #print k

    print len(pos)

    #Creating list of pixel locations
    dictlist=[]
    temp=[]
    for key in range(0, len(pos)):
        temp = pos[key]
        dictlist.append(temp)

    #print len(dictlist)

    kdtree = spatial.KDTree(dictlist)
    other = kdtree
    #print type(dictlist)

    k=0
    print "made kd tree"

    #Takes the nodes of each crack pixel in a KD-tree and finds the nearest
neighbor for th value specified
    #This is a much faster way of pairing nodes that are close to each other,
rather than looking at each
    #node and comparing it to each other node
    begin=time.time()
    try:
        pairs=kdtree.query_pairs(7)
    except MemoryError:
        return

    #pairs=kdtree.query_ball_tree(other,r=40)
    #print type(pairs)

    end=time.time()
    print "step time: " + str(int(end-begin))+" seconds"

    #print type(pairs)
    #print len(pairs)
    print "made pairs"

    #adds edges between nodes that are close enough
    try:
        G.add_edges_from(pairs)
    except MemoryError:
        return

    #creates minimum spanning tree with kruskals algorithm in order to remove
unneccessary edges
    #This is done to save memory and runtime
    s=nx.Graph()
```

```
    s=nx.algorithms.mst.minimum_spanning_tree(G)

    print "added edges"

    #connected crack pixels are drawn on a graph
    nx.draw_networkx(s,pos,False,node_size=0)
    #plt.show()

    cv2.waitKey(0)
    cv2.destroyAllWindows()
    h=nx.Graph()
    h=s

    #removes nodes with only one connection, reduces errors
    outdeg = h.degree()
    to_remove=[n for n in outdeg if outdeg[n] ==1]
    h.remove_nodes_from(to_remove)

    plt.figure(figsize=(16,12))
    nx.draw_networkx(h,pos,False,node_size=0)
    plt.axis('off')
    plt.subplots_adjust(left=0, bottom=0, right=1, top=1, wspace=0, hspace=0)
    plt.xlim((-1600,0))
    plt.ylim((0,1200))

    #Saves Graph as a jpg
    plt.savefig('C:/Users/Pat/documents/Google
Drive/temp/algocrack/'+str(name)+str(file)+'.jpg', dpi=100,pad_inches=0)
    #plt.show()
    plt.close()

    return(None)
```

```
#compare5.py
#compares Pixels
def compare(file,name):

    import networkx as nx
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    import matplotlib

    import numpy as np

    import cv2

    import sys
    import time

    from PIL import Image

    #Loads File crack file developed by algorithm from called arguments as
well as
    #hand identified crack file
    im1=cv2.imread("C:/Users/Pat/Documents/Google
Drive/temp/Cracks2/"+str(file)+".jpg",cv2.CV_LOAD_IMAGE_GRAYSCALE)
    im2=cv2.imread("C:/Users/Pat/Documents/Google
Drive/temp/algocrack/"+str(name)+str(file)+".jpg",cv2.CV_LOAD_IMAGE_GRAYSCALE
)
    print "test"

    #checks files are loaded
    #cv2.namedWindow('dst_rt')
    #cv2.imshow('dst_rt', im1)
    #cv2.waitKey(0)
    #cv2.destroyAllWindows()

    #image sizes
    width=1200
    height=1600

    #checks files are loaded
    #cv2.resizeWindow('dst_rt', window_width, window_height)
    #cv2.resizeWindow('dst_rt', window_width, window_height)
    #cv2.waitKey(0)
    #cv2.destroyAllWindows()
    #cv2.resizeWindow('dst_rt', window_width, window_height)
    #plt.imshow(im2)
    #cv2.waitKey(0)
    #cv2.destroyAllWindows()

    span=15#distance to check pixels to check for truth,
    #if less, crack pixel is a false positive or false negative
    foundcrackpixels=0
    count=0
    truecrack=False
    #begins iteration of checking every ground truth crack pixel against the
corresponding pixel
    #of the algorithm image within the specified range.  The loop counts all
true positives and
```

```
    #all positives possible found by the ground truth.

    begin=time.time()
    for x in range(0,width-1):
        for y in range(0,height-1):
            if im1[x,y]<200:
                foundcrackpixels+=1
                for x2 in range(-span,span):
                    for y2 in (-span,span):
                        if (x-x2)<1 or (x+x2)<1 or (x+x2)>width-1 or (x-
x2)>width-1:
                            break
                        if (y-y2)<1 or (y+y2)<1 or (y+y2)>height-1 or (y-
y2)>height-1:
                            continue
                        if im2[x-x2,y-y2]<200:
                            truecrack=True
                if truecrack==True:
                    count+=1
                truecrack=False
        if x%10 == 0:
            print "......"+str(int(100*(float(x)/float(width))))+"%\r",

    print "First Run Completed"
    end=time.time()
    print "step time: " + str(int(end-begin))+" seconds"

    #begins iteration of checking every algorithm truth crack pixel against
the corresponding pixel
    #of the true crack image within the specified range.  The loop counts all
false negatives
    FalseNegative=0
    begin=time.time()
    for x in range(0,width-1):
        for y in range(0,height-1):
            if im2[x,y]<200:

                for x2 in range(-span,span):
                    for y2 in (-span,span):
                        if (x-x2)<1 or (x+x2)<1 or (x+x2)>width-1 or (x-
x2)>width-1:
                            break
                        if (y-y2)<1 or (y+y2)<1 or (y+y2)>height-1 or (y-
y2)>height-1:
                            continue
                        if im1[x-x2,y-y2]<200:
                            truecrack=True
                if truecrack==False:
                    FalseNegative+=1
                truecrack=False
        if x%10 == 0:
            print "......"+str(int(100*(float(x)/float(width))))+"%\r",
    end=time.time()
    print "step time: " + str(int(end-begin))+" seconds"

    #makes sure the script doesnt throw an error for dividing by zero
    if foundcrackpixels!=0:
```

```
        Recall=(float(count)/float(foundcrackpixels))
    else:
        Recall=0
    #print "true crack pixels = "+str(count)
    if count==0 and foundcrackpixels==0:
        Recall=1

    if FalseNegative!=0 and count!=0:
        Precision = float(count)/float(count+FalseNegative)
    else:
        Precision=0
    if count==0 and FalseNegative==0:
        Precision=1

    #print "false crack pixels ="+str(foundcrackpixels-count)
    #False Positive rate
    if Recall!=0 and Precision!=0:
        Fmeasure= 2*(Precision*Recall/(Precision+Recall))
    else:
        Fmeasure=0

    #Prints results of each image to a csv file that will be opened in excel
    #records the file number, Precision, Recall, and F-Measure
    print "File number: " + str(file)
    print "Precision =" + "{:.1%}".format(Precision)
    print "Recall =" + "{:.1%}".format(Recall)
    print "F-Measure =" + "{:.1%}".format(Fmeasure)

    return
"{:.1%}".format(Precision)+'\t'+"{:.1%}".format(Recall)+'\t'+"{:.1%}".format(
Fmeasure)
```

Bibliography

Clark, Richard M. 2000.*Uninhabited Combat Aerial Vehicles: Airpower by the People, for the People, but Not with the People*: 14.

Cline, Gregory D., Mohamed Y. Shahin, and Jeffrey A. Burkhalter. 2003. "Automated Data Collection for Pavement Condition Index Survey.".

Coletta, P. E. 1987. "Patrick N.L. Bellinger and U.S. Naval Aviation." , 64: University Press of America.

Davis, Lynn E., Michael J. McNerney, James Chow, Thomas Hamilton, Sarah Harting, and Daniel Byman. 2014.*Armed and Dangerous? UAVs and US Security*: 11.

Elkins, Gary E., Gonzalo R. Rada, Jonathan L. Groeger, and Beth Visintine. 2013. *
Pavement Remaining Service Interval Implementation Guidelines* . Research, Development, and Technology: U.S. Department of Transportation.

Federal Highway Administration. 2011. *Pavement Management Primer*.

Federal Aviation Administration. 2015. Press release – DOT and FAA propose new rules for small unmanned aircraft systems. , 15 Feb 2015, 2015.

Galehouse, Larry, James S. Moulthrop, and R. Gary Hicks. 2011. *Principles of Pavement Preservation: Definitions, Benefits, Issues, and Barriers* U.S. Department of Transportation Federal Highway Administration.

Giakoumis, Ioannis, Nikos Nikolaidis, and Ioannis Pitas. 2006. "Digital Image Processing Techniques for the Detection and Removal of Cracks in Digitized Paintings." *Image Processing, IEEE Transactions On* 15 (1): 179.

Haas, Ralph. 2001. "Reinventing the (Pavement Management) Wheel.".

Keane, John F. and Stephen S. Carr. 2013. "A Brief History of Early Unmanned Aircraft." *Johns Hopkins APL Technical Digest* 32 (3): 558.

Laibson, David. 1997. "Golden Eggs and Hyperbolic Discounting." *The Quarterly Journal of Economics*: 443-477.

Laurent, John, Jean François Hébert, Daniel Lefebvre, and Yves Savard. 2012. "Using 3D Laser Profiling Sensors for the Automated Measurement of Road Surface Conditions (Ruts, Macro-Texture, Raveling, Cracks)." : 1.

Losey, Stephen. 2014. "AF Secretary: 18,700 More Airmen Cuts before it's Over." *Air Force Times*, 19 May 2014.

Nissen, Mark R. 1993. "Image Blur." *Unknown* 1.

Papert, Seymour. 1966. "The Summer Vision Project." .

Pawlyk, Oriana. 2014. "Vice Chief to Airmen: How do we Cut Costs?" *Air Force Times*, 30 Sep 2014.

Reed, John. 2013. "Predator Drones 'Useless' in most Wars, Top Air Force General Says." *Foreign Policy*, September 19, 2013.

Sayers, Michael W., Thomas D. Gillespie, and AV Queiroz. 1986.*The International Road Roughness Experiment.Establishing Correlation and a Calibration Standard for Measurements*: 26.

Shahin, Mo Y. 2007. "Pavement Management for Airports, Roads, and Parking Lots." In , 94: Springer.

Siebert, Sebastian and Jochen Teizer. 2014. "Mobile 3D Mapping for Surveying Earthwork Projects using an Unmanned Aerial Vehicle (UAV) System." *Automation in Construction* 41: 1-14.

Szeliski, R. 2010. "Computer Vision: Algorithms and Applications." In , 3: Springer.

Tatham, Peter. 2009. *An Investigation into the Suitability of the use of Unmanned Aerial Vehicle Systems (UAVS) to Support the Initial Needs Assessment Process in Rapid Onset Humanitarian Disasters*. Vol. 13. doi:10.1504/IJRAM.2009.026391. http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=44477345&site=ehost-live.

Ting, Kai Ming. 2010. "Precision and Recall." In *Encyclopedia of Machine Learning*, 781: Springer.

United States Department of Defense. 2004. *UFC 3-270-08 Pavement Maintenance Management*.

Zou, Qin, Yu Cao, Qingquan Li, Qingzhou Mao, and Song Wang. 2012. "CrackTree: Automatic Crack Detection from Pavement Images." *Pattern Recognition Letters* 33 (3): 231.

Vita

Captain Patrick J. Grandsaert graduated from Bellarmine College Preparatory in San Jose, California. He was appointed to the United States Air Force Academy in Colorado Springs, Colorado where he graduated with a Bachelor of Science degree in Civil Engineering and a commission as a second lieutenant in the Unites States Air Force in 2009.

His first assignment was at Laughlin AFB as a student in Undergraduate Pilot Training in August 2009. In June 2010, he was assigned to the 50th Civil Engineer Squadron, Schriever AFB, Colorado where he served in jobs ranging from overseeing the maintenance and repair of all base infrastructure as the acting Operations Flight Chief, overseeing SABER construction, as well as planning out year infrastructure modernization projects. In August 2013, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation, he will be assigned to the 8th Civil Engineer Squadron, Kunsan AFB, Korea as the Chief of Project Management.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
27-3-2015	Master's Thesis	13-8-2013 - 27-3-2015

4. TITLE AND SUBTITLE
Integrating Pavement Crack Detection And Analysis Using Autonomous Unmanned Aerial Vehicle Imagery

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Grandsaert, Patrick J., Capt USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENV)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENV-MS-15-M-195

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A: Approved for Public Release Distribution Unlimited

13. SUPPLEMENTARY NOTES

Efficient, reliable data is necessary to make informed decisions on how to best manage aging road assets. This research explores a new method to automate the collection, processing, and analysis of transportation networks using Unmanned Aerial Vehicles and Computer Vision technology. While there are current methodologies to accomplish road assessment manually and semi-autonomously, this research is a proof of concept to obtain the road assessment faster and cheaper with a vision for little to no human interaction required. This research evaluates the strengths of applying UAV technology to pavement assessments and identifies where further work is needed. Furthermore, it validates using UAVs as a viable way forward for collecting pavement information to aid asset managers in sustaining aging road assets.

The system was able to capture road photos suitable for semi automated Pavement Condition Index (PCI) processing, however the algorithm resulted in a maximum F-Measure of 40%. This result is low and indicates the algorithm is not sufficient for fully automated PCI classification. Accurately detecting road defects using computer vision remains a challenging problem for future research. However, using Autonomous UAVs to collect the data is a viable avenue for data collection, theoretically faster than current methods at freeway speeds.

15. SUBJECT TERMS
Computer Vision, Unmanned Aerial Vehicles, Autonomous, Pavement, PCI

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maj Vhance, V. Valencia, AFIT/ENV
U	U	U	UU	68	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, x 4826 vhance.valencia@afit.edu